The Finish Dial

The top singular directions of an RL weight update
set how often a language model finishes its answer

The Invariant Swarm

monitor.invariant.team · November 2026

Abstract. Reinforcement learning with verifiable rewards improves a base model at mathematics, and a few singular directions of the weight change are known to carry most of that gain. We ask what those directions do to an answer. In Olmo 3 7B we add the top 8 singular directions per weight matrix of (RL model − base) to the base, at a size we vary from −1 to 3. On 328 hard competition problems the size sets how often the model finishes inside its token budget: answers cut off at the cap go from 96 to 77 to 46 to 5 as the size goes from −1 to 0 to 1 to 2. A random edit with the same singular values is flat at the four sizes tested, and removing the directions from the RL model returns it to the base's level (25 to 90 cut-off answers). At size 1.5, the best of seven sizes on these problems, the edited base matches the RL model's mean score over four samples (200.3 against 200.0 of 328) and trails it under a vote over the four (214 against 221). The edit is read from the RL model's weights, so it does not replace the RL run; it describes what the top directions of this run do. They set how often the model commits to an answer, and from the base up to size 1.5 the answers it commits to are right as often as before.

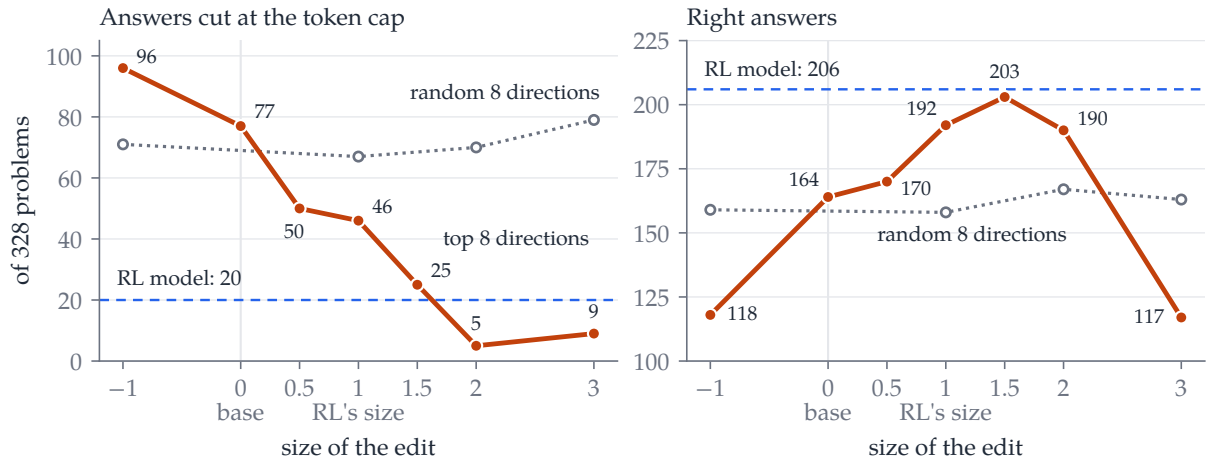


Figure 1: The dial. The base model plus *size* × the top 8 singular directions per matrix of (RL − base), on 328 hard problems, one sample per point. **Left:** answers cut at the 16,384-token cap. **Right:** right answers. Dotted: the same edit with random directions. Dashed: the RL model, one sample at the same seed (over five runs it has 190 to 209 right and 20 to 35 cut). Repeated runs of one model range over 10 to 19 problems.

1 Introduction

A model trained with reinforcement learning from verifiable rewards (RLVR) [1, 2] differs from its base by a small weight change, and that change is close to low rank: one or a few singular directions per matrix recover most of the accuracy gain [3–5]. We ask what those directions do to an answer.

An answer to a hard problem can fail in two ways: the model writes until the token cap and never commits (a *cap hit*), or it finishes with a wrong answer. So accuracy is, to within a few answers, a product: right = finished × right among finished, and an edit can move either factor. We scale the low-rank part of the RL update up and down and read both (Figure 1). The size of the edit moves the first factor over almost its whole range, in both directions, and a random edit of the same size moves neither. The second factor, which we call precision, is measured on a different set of answers at each size, so we also count

right answers on the fixed set of problems the base already finishes. The claim we defend is that the top directions set finishing, monotonically up to size 2 and specifically, not that they leave correctness untouched: they also raise it, by less.

2 Setup

Models. All models are 7B checkpoints of the Olmo 3 family [6] and share one base: Olmo 3 Base; *RL*, the Olmo 3.1 RL-Zero Math checkpoint, trained from the base with RLVR and no demonstrations; and *Think-SFT*, the Olmo 3 Think SFT checkpoint, trained from the base by supervised fine-tuning on reasoning traces, with no reward.

Problems. 328 competition problems (AIME 2022–2025, AMC, HMMT, OlympiadBench): a pool of 400 minus the 72 we found, exactly or nearly, in the RL model's training prompts. We also report MATH-500, the 500-problem subset [7] of the MATH test set [8]. We sample at temperature 1.0 with at most 16,384 new tokens; an answer that reaches the limit is a cap hit. An answer is right if its last boxed expression or `Answer:` line is equivalent to the reference; a few cut answers are right this way (3 of the base's 77).

Prompt. Fixing the first tokens of the response recovers much of the RL gain on this base [9]. The base, every edit of it and Think-SFT therefore start from a three-token prefilled opening ending in "Okay", which we call the cue (ours is `` `` line!\n\nOkay'' ``; that paper uses a two-token one). Every gain below is on top of the cue. The RL model runs without it; with it, it scores the same (205 right and 25 cap hits against 206 and 20, one sample each).

The edit. Let $D = W_{\mathrm{RL}} - W_{\mathrm{base}}$ for each of the 224 attention and MLP projection matrices (7 per layer, 32 layers), and $D_8 = U_8 \Sigma_8 V_8^\top$ its top 8 singular directions, which hold 9.8 % of $\|D\|^2$. The edited model is $W_{\mathrm{base}} + s\, D_8$, with size s. The control replaces U_8 and V_8 by random orthonormal matrices and keeps Σ_8, so it has the same norm in every matrix. Intervals are 95 % paired bootstrap intervals over problems, in percentage points of 328.

3 Results

3.1 The size of the edit sets how often the model finishes

Cap hits go 96, 77, 50, 46, 25, 5 of 328 as the size goes −1, 0, 0.5, 1, 1.5, 2 (Figure 1, left), and 9 at size 3; the paired interval of every step up to size 2 except 0.5 to 1 excludes zero. Subtracting the directions makes the base finish less often than it does untouched; twice the RL amount leaves almost nothing unfinished. The random edit stays between 67 and 79 cap hits at sizes −1, 1, 2 and 3. Mean answer length changes little (9,344 tokens for the base, 8,916 at size 1.5), so the edit does not shorten answers across the board; it ends the ones that would have run to the cap.

Right answers rise and then fall (Figure 1, right): 164 for the base, 192 at size 1, 203 at size 1.5, 190 at size 2, and 117 at size 3, where the model breaks. At size 1 the edit is +10.4 points [5.8, 14.9] above the random edit of the same size, and at size 2 it is +7.0 [2.7, 11.3] above.

Between sizes 0 and 1.5 the share of finished answers that are right stays between 60 and 67 % (64.1 % for the base, 66.3 % at 1.5). This is not because the edit leaves correctness alone. At size 1.5, 64 of the base's 77 cap hits now finish, and only 19 of them are right: they are the harder problems, so an edit that changed only stopping would lower precision. It holds because right answers on the 251 problems the base already finishes rise from 161 to 184 (random edit: between 149 and 158 at its four sizes, below the base; RL model: 187). Of the 39 answers gained at size 1.5, 16 are on problems the base ran to the cap and 23 on problems it already finished. Outside this range precision falls: to 50.9 % at size −1, 58.8 % at size 2 and 36.4 % at size 3. Figure 2 shows the same runs with the two factors as axes.

3.2 Without the directions, the RL model finishes no more often than the base

To test whether the RL model needs the directions, we subtract D_8 from the RL model itself and run both with the cue (one sample each, different seeds). Cap hits go from 25 to 90 (+19.8 points [15.2, 24.4]), back

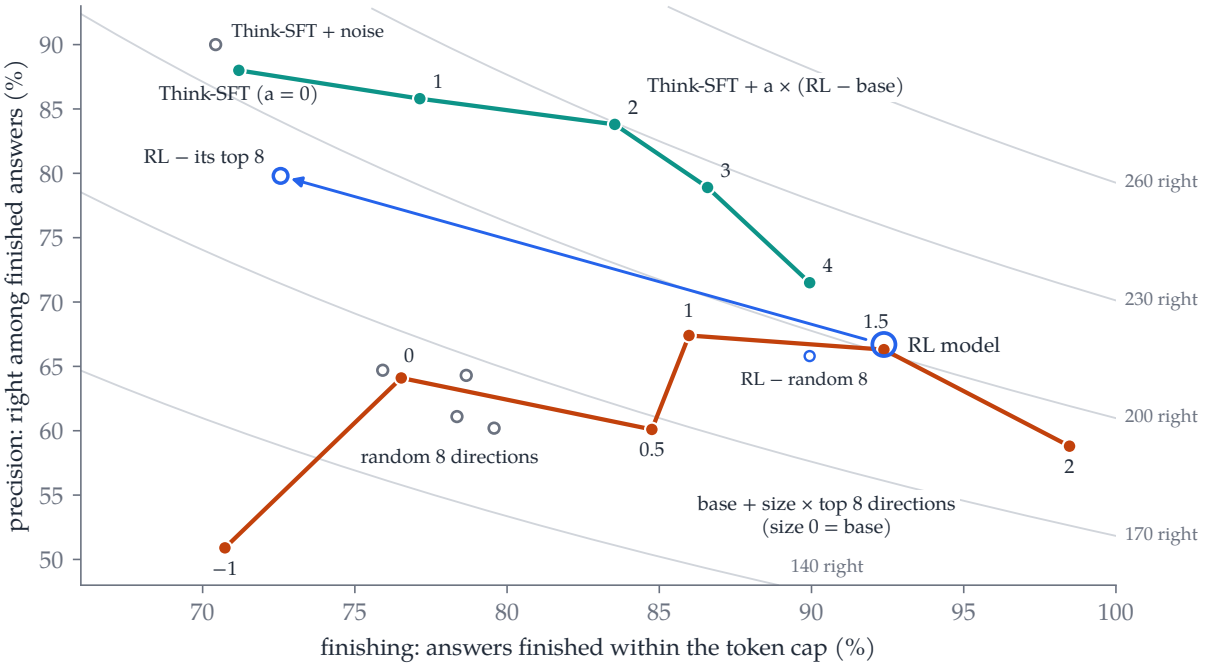


Figure 2: Finishing and precision, one point per model on the 328 problems. Grey curves join points with the same number of right answers; up and to the right is better. **Orange:** the dial on the base moves mostly sideways between sizes 0 and 1.5, where it lands on the RL model, and loses precision at −1 and 2 (size 3, at 97 % finished and 36 % precision, is off the plot). **Blue:** removing the same 8 directions from the RL model moves it left and up: it finishes fewer, easier problems. Removing 8 random ones (small blue ring) does not. **Green:** adding a × the whole RL update to the imitation-trained model also buys finishing, and pays in precision. Grey rings are the random-direction and noise controls. The RL model is run with the cue here (205 right, 25 cut). Orange and blue points are one sample; green points are means of one to four.

to the level of the base (77). Subtracting a random rank-8 matrix with the same singular values leaves 33. Right answers fall from 205 to 190 (−4.6 points [−8.2, −1.2]), within noise of the random removal (195): what the model still finishes it gets right 79.8 % of the time, on an easier, self-selected set, because the long, hard answers are the ones that no longer end.

3.3 At size 1.5 the edited base matches the RL model's mean score

Size 1.5 was the best of seven sizes on one sample of the 328, so we drew again (Table 1). Over four samples the edited base averages 200.3 right and the RL model 200.0, a difference of +0.1 points [−1.7, 1.8]; the base with the cue averages 158.5. The first of the four samples reuses the seeds of the original run; the three new draws give 198, 199 and 197. On MATH-500, which played no part in choosing the size, one sample of the edit scores 456 of 500, equal to the lowest of the RL model's four samples (456 to 464, mean 460.0). Under a plurality vote over the four samples the edit trails the RL model, 214 against 221 (−2.1 points [−5.2, 0.6]).

	328 hard problems, four samples			MATH-500
Model	right, mean (each)	cap hits, mean	vote over 4	right of 500
Base + cue	158.5 (161, 161, 152, 160)	71.5	192	405–418
Base + cue + 1.5 × D_8	200.3 (207, 198, 199, 197)	19.0	214	456
RL model	200.0 (207, 190, 209, 194)	28.5	221	456–464

Table 1: The edit at size 1.5 against the RL model. MATH-500 ranges are four samples; the edit has one.

3.4 Which rewards write the dial

The Olmo 3 release has RL-Zero checkpoints trained from the same base on four rewards: math, code, instruction following and a general mix. We built the top-8 edit from each, at size 1 (Table 2). Five of the six cut cap hits, the instruction-following one included. Only the general one does not, and its edit has half the norm of the math edit, so that null is not size-matched. What separates the rewards is the score: Math 3.1 raises it most (192), Math 3.0 and the code checkpoints follow within noise of each other, and instruction following cuts cap hits without raising it (155, -2.7 points $[-7.3, 1.5]$ against the base).

We measure how much two checkpoints share a subspace as $\|V_a^\top V_b\|_F^2/8$ for their top-8 input directions, averaged over matrices: 1 for identical subspaces, 0.002 for random ones, and 0.04 between the math update and the base matrices' own top 8. Two math checkpoints overlap by 0.48, math and code by 0.26, and math with instruction following by 0.03. The instruction-following edit still cuts cap hits, so finishing is not confined to one subspace. Rank 1 already carries most of the effect: at size 1, ranks 1, 8 and 64 give 181, 192 and 200 right (base 164) and 51, 46 and 42 cap hits; rank 64 is above rank 1 by $+5.8$ points $[1.8, 10.1]$.

Source of the 8 directions	Right of 328	Cap hits
None (base + cue)	164	77
RL-Zero Math 3.1 (the RL model)	192	46
RL-Zero Math 3.0	182	43
RL-Zero Code 3.1	178	41
RL-Zero Code 3.0	173	23
RL-Zero Instruction following 3.0	155	41
RL-Zero General 3.0	164	88

Table 2: The base plus the top 8 directions of six RL checkpoints, size 1, one sample each.

3.5 The same trade in a model trained by imitation

Think-SFT sits at the opposite corner of Figure 2: 88.0 % of its finished answers are right, but it is cut at the cap on 94.5 of 328 problems. Its weight change from the base is about 114 times larger than the RL update and orthogonal to it (cosine -0.002). We add a times the whole RL update to it. Cap hits fall with every step (94.5, 75, 54, 44, 33 for $a = 0$ to 4) and precision falls with them (88.0, 85.8, 83.8, 78.9, 71.5 %), so the score peaks in between: 231.5 of 328 at $a = 2$, above Think-SFT (209.0) and the RL model (200.0). Gaussian noise of the same intended norm changes nothing (209 right, 97 cap hits). That a sum of an imitation model and an RL update can beat both is known [10]; a trade of cap hits for precision is consistent with the optimum being interior. Here the edit is the full update, not its top 8 directions, and $a = 1$, 3 and 4 are one sample.

4 Related work

Low-rank RL updates. The top singular direction of an RLVR update recovers nearly all of its accuracy gain [3–5] and evolves near-linearly over training, so later checkpoints can be extrapolated from early ones [3, 4]; RLVR updates avoid the principal directions that supervised fine-tuning targets [11]. These works report accuracy. We report what the directions do to an answer, with a random-direction control and a removal test. **Scaling weight differences.** Task vectors can be added, negated and scaled [12]; scaling a fine-tuning update past 1 can improve a model [13]; a weight direction between two code-RL policies controls resource-limit failures [14]; and an edit of a few attention heads changes how long a reasoning model thinks [15]. Reasoning updates transfer between models [16] and combine with imitation updates [10]. **What RL adds.** RLVR raises accuracy mostly on problems the base can already solve in some samples [17, 18]; part of its reported out-of-domain gain is answer format [19]; and an opening cue recovers much of it [9]. We add a weight-space account of one more part: finishing.

5 Limitations

- The dial is read from the RL model's weights. It explains the RL update; it does not replace the RL run.
- One model family, one size (7B), mathematics. Each point of the dial is one sample of 328 problems, and repeated runs of one model range over 10 to 19 problems, so only the large contrasts are resolved.
- Size 1.5 was chosen on the 328. The new samples are on the same problems; MATH-500 is the only other set.
- The random control was run at sizes -1, 1, 2 and 3, not at 0.5 or 1.5.
- One token cap. At a larger cap fewer answers are cut, and we have not measured the dial there.
- The weights are stored in bfloat16, and at these sizes much of the stored edit is rounding noise: at size 1.5 the edit is 0.11 % of the weight norm as intended and 0.19 % as stored. The random control is rounded the same way and stays flat, so the effect is not the rounding.

6 Conclusion

In this model the best-known part of the RL update, its top singular directions, works as a control on committing to an answer. It can be turned up past what RL chose, turned down below the base, and taken out of the RL model, and the number of unfinished answers follows each time. Finishing can be moved over most of its range while precision stays within a few points, which suggests reporting the two separately: a benchmark score at a fixed token cap mixes how well a model reasons with how readily it stops.

Reproducibility. Code, the metrics of every run and the script that draws both figures are in the project repository under `studies/rlneurons/`; each number in this paper is listed with its source in the file `paper/facts.json` of that folder.

References

[1] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models. *arXiv preprint arXiv:2402.03300*, 2024.

[2] DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, et al. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. *arXiv preprint arXiv:2501.12948*, 2025.

[3] Yuchen Cai, Ding Cao, Xin Xu, Zijun Yao, Yuqing Huang, Zhenyu Tan, Benyi Zhang, Guangzhong Sun, Guiquan Liu, and Junfeng Fang. On Predictability of Reinforcement Learning Dynamics for Large Language Models. *arXiv preprint arXiv:2510.00553*, 2025.

[4] Zhepei Wei, Xinyu Zhu, Wei-Lin Chen, Chengsong Huang, Jiaxin Huang, and Yu Meng. You Only Need Minimal RLVR Training: Extrapolating LLMs via Rank-1 Trajectories. *arXiv preprint arXiv:2605.21468*, 2026.

[5] Hao Ye, Jisheng Dang, Junfeng Fang, Bimei Wang, Yizhou Zhang, Ning Lv, Wencan Zhang, Hong Peng, Bin Hu, and Tat-Seng Chua. On the Implicit Reward Overfitting and the Low-rank Dynamics in RLVR. *arXiv preprint arXiv:2605.06523*, 2026.

[6] Team Olmo, Allyson Ettinger, Amanda Bertsch, Bailey Kuehl, David Graham, David Heineman, Dirk Groeneveld, Faeze Brahman, Finbarr Timbers, Hamish Ivison, et al. Olmo 3. *arXiv preprint arXiv:2512.13961*, 2025.

[7] Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's Verify Step by Step. *arXiv preprint arXiv:2305.20050*, 2023.

[8] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring Mathematical Problem Solving With the MATH Dataset. *arXiv preprint arXiv:2103.03874*, 2021.

[9] Sophie L. Wang, Amil Dravid, Rulin Shao, Kevin Farhat, Sewon Min, and Alexei A. Efros. Base Models Can Reason By Taking a Cue From Training Data. *arXiv preprint arXiv:2610.06851*, 2026.

[10] Chaohao Yuan, Chenghao Xiao, Yu Rong, Hong Cheng, and Long-Kai Huang. Decouple before Integration: Test-time Synthesis of SFT and RLVR Task Vectors. *arXiv preprint arXiv:2605.00610*, 2026.

[11] Hanqing Zhu, Zhenyu Zhang, Hanxian Huang, DiJia Su, Zechun Liu, Jiawei Zhao, Igor Fedorov, Hamed Pirsiavash, Zhizhou Sha, Jinwon Lee, David Z. Pan, Zhangyang Wang, Yuandong Tian, and Kai Sheng Tai. The Path Not Taken: RLVR Provably Learns Off the Principals. *arXiv preprint arXiv:2511.08567*, 2025.

[12] Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing Models with Task Arithmetic. *arXiv preprint arXiv:2212.04089*, 2022.

[13] Chujie Zheng, Ziqi Wang, Heng Ji, Minlie Huang, and Nanyun Peng. Model Extrapolation Expedites Alignment. *arXiv preprint arXiv:2404.16792*, 2024.

[14] Kunhao Zheng, Pierre Chambon, Juliette Decugis, Jonas Gehring, Taco Cohen, Benjamin Negrevergne, and Gabriel Synnaeve.

Correctness vs Efficiency: Behavior Control with Weight-Space Extrapolation in Code RL. *arXiv preprint arXiv:2605.28751*, 2026.

[15] Chung-En Sun, Ge Yan, and Tsui-Wei Weng. ThinkEdit: Interpretable Weight Editing to Mitigate Overly Short Thinking in Reasoning Models. *arXiv preprint arXiv:2503.22048*, 2025.

[16] Mohammad Zbeeb, Hasan Abed Al Kader Hammoud, and Bernard Ghanem. Reasoning Vectors: Transferring Chain-of-Thought Capabilities via Task Arithmetic. *arXiv preprint arXiv:2509.01363*, 2025.

[17] Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Yang Yue, Shiji Song, and Gao Huang. Does Reinforcement Learning Really Incentivize Reasoning Capacity in LLMs Beyond the Base Model? *arXiv preprint arXiv:2504.13837*, 2025.

[18] Minwu Kim, Anubhav Shrestha, Safal Shrestha, Aadim Nepal, and Keith Ross. Reinforcement Learning vs. Distillation: Understanding Accuracy and Capability in LLM Reasoning. *arXiv preprint arXiv:2505.14216*, 2025.

[19] Zongye Lyu. Same Answer, New Box: How Much of RLVR's Out-of-Domain Gain Is Answer Format? *arXiv preprint arXiv:2610.20639*, 2026.